Containers

COSC349—Cloud Computing Architecture

David Eyers

Learning objectives

- Define what a (software) **container** is
- Give two benefits and two downsides of containers compared to full (hardware) virtual machines
- Explain how a container framework like Docker **optimises handling filesystems** for its lightweight VMs
- Describe the role of online sites like **Docker Hub** in helping software developers use containers

Lightweight virtualisation of software

- We have traced evolution of virtualisation
  - Complete but non-real-time simulation
  - Fast, but expensive full-machine virtualisation
  - OS-level virtualisation of userspaces
- Also discussed features of operating systems like CoW filesystems that support snapshots & rapid cloning
- This is all about how to **run** VMs though, not about how to efficiently **manage** the software **within** the VM

Compare using Vagrant to using VirtualBox

- You have seen how both tools work in the lab exercises
  - VirtualBox provides a GUI (for VMs too): configure your VMs
  - Vagrant focuses instead on the software running on your VMs

- Vagrant accelerates developer-focused use of VMs:
  - Each VM's "hardware" gets a sane default configuration
  - Vagrant box files only download once
  - SSH interface facilitates convenient developer access
  - Context-based VM selection based on working directory

Software container frameworks, e.g. Docker

- Container is a ‘standard’ unit of OS-level virtualisation
  - Analogous to physical multimodal shipping container (ISO 668)
  - Works well in a Linux context (software licences not required)

- Usually containers run within OS-level virtualisation
- Attention paid to the container management API/CLI
  - *i.e.*, App. Programming Interface & Command Line Interface
- Container framework helps manage OS resources
  - particularly disk, RAM and network

RAM optimisation for containers

- RAM is an expensive resource for virtualisation
- Unlike CPU, can't effectively time share: significant performance drop to swap data between RAM & disk
  - Would involve lots of reads and writes to disk

- Containers help by **avoiding duplication of OS kernel**
- Within VMs, containers can memory map one instance of **each shared library** for further de-duplication
  - but this breaks when multiple versions of a library are used

Filesystem management for containers

- Hard-disks in full hardware virtualisation typically **appear opaque** to the host (but recall exceptions)
  - Wasteful when guests disks are very similar but not identical
  - Situation arises when VMs deployed from common template
- VirtualBox supports cloning of disks and JIT allocation
  - However the filesystem data is still opaque to the host
- Also, the filesystem is effective to share data with host
  - VirtualBox shared folders used by Vagrant to mount `/vagrant`
- We'll return to this topic, using Docker as an example…

Introducing Docker and its aims

- Docker is a popular **container framework**
  - Provides tools to unify a collection of Linux technologies
    - There are many alternatives—most achieve similar effects
    - Windows can now host Windows containers—we won't explore this
  - Over time Docker has replaced some of the technologies it used with versions developed by the Docker team directly
- Docker aims to make OS-level virtualisation **usable**
  - Facilitates flexible targeting both on-premises and cloud-hosted
- Docker is also an **online ecosystem**
  - Docker can be used privately, but often uses public resources

Docker on macOS and Windows

- Docker uses features within the Linux kernel
    - So using macOS or Windows as a host first needs a Linux kernel
- Older approach—**Docker Toolbox**
    - Similar effect to running Ubuntu VM to then run containers
        - VirtualBox usually used as the VMM for the Linux VM hosting Docker
- Newer approach—**Docker Desktop**
    - Allows use of recent, advanced OS features, *e.g.*,
        - Uses hypervisor framework on macOS and Windows
        - Can use APFS on Apple for Docker image storage

Container disk handing—Docker images

- Vagrant boxes are typical, cached starting points
  - Your VMs might start with Ubuntu, then shell provision software
  - VMs disk images are then opaquely different to VMM, though
- Docker images—virtual hard disks—are **built from layers**
  - Layers store sets of files and directories; identified by hash
  - Layers might be: (1) Ubuntu; (2) + web server; (3) + your app.
  - Layer stored as delta from parent: can be cached and shared
- Host drivers may allow host to see guest filesystem if host filesystem can isolate directory subtrees

Docker storage drivers

- **Union filesystems**: overlay multiple directories
  - e.*g*., read-write filesystem overlaid over read-only filesystem
    - Files get "copied up" for writing at read-write layer on demand
    - Use "white out" files to "delete" files from lower layers
  - AUFS—Advanced multi-layered Unification Filesystem
    - Unfortunately AUFS is not in the mainline Linux kernel
  - overlayfs (overlay)—simpler than AUFS; mainline kernel
- **CoW filesystems** if your host has them—BTRFS, ZFS, etc.
- Storage drivers can potentially work at block level, too

Sharing files between containers / host

- VMs see VirtualBox shared folders as network drives
  - Requires VMs to install the Guest Extensions
  - (Vagrant boxes typically already include the Guest Extensions)
- Docker can do sharing more directly:
  - Containers can mount host filesystems (same OS kernel)
- Docker **bind mounts**—one folder mounted twice
  - Inside mount used by container; outside mount is on host
- Docker **volumes**—Docker setups up bind mount for you
  - Preferred: host-side bind mount doesn't need explicit config.

Software ecosystems

- Ecosystems lift software functionality beyond tool itself:
  - GitHub's impact on git
  - Vagrant Cloud's benefits over directly using VirtualBox
- DockerHub is a public **sharing site for Docker images**
  - … well, specifically layers of images
  - Since anyone can share, do **consider malware risks**
    - Use officially-certified containers wherever possible
- Docker tools let you push content to DockerHub
  - Also can create ‘**Automated builds**’; runs build in the cloud